Linear logic and elementary time

ACADEMIC PRESS

Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Information and Computation 183 (2003) 123–137

Information and Computation

www.elsevier.com/locate/ic

Linear logic and elementary time

Vincent Danos[a,*] and Jean-Baptiste Joinet[b]

[a] *CNRS and Université Paris 7, Paris, France*
[b] *Université Paris 1 and CNRS, Paris, France*

Received 11 January 2000; revised 12 March 2001

Abstract

A subsystem of linear logic, *elementary linear logic,* is defined and shown to represent exactly *elementary recursive functions.* Its choicest part consists in reducing the deductive power of the exponential, also known as the "bang," which, in linear logic, is in charge of controlling duplication in the cut-elimination process.

1. Introduction

Think of elementary linear logic as an idealized functional programming language with a severe typing mechanism. Definition by recursion is, of course, forbidden, but some sort of iteration still is possible and the purpose of this paper is to show that enough computing power remains so that elementary recursive functions can be implemented. Actually, the whole paper can be considered an exercise in programming elegantly with a rather desolate language.

To zero in on an interesting class of functions, one usually tries to weaken in the given logic whatever corresponds to induction or iteration. Here we follow a different strand, rather specific to the linear logic decomposition of the implication as $!A \multimap B$, by fiddling with the rules handling "!". The standard rules are enough to embed the full power of intuitionistic computations. So the game is to find a sensible way to make them harder to use than in full linear logic. There are a few obvious candidates which we present below and ELL stands out as the simplest.

By proving it represents elementary time we surely learn something about the logic; but, does the logic teach us anything about elementary time? It seems we merely are giving yet another

* Corresponding author.
E-mail address: danos@logique.jussieu.fr (V. Danos).

0890-5401/03/$ - see front matter
doi:10.1016/S0890-5401(03)00010-5

language defining that complexity class. But the language is just not any language. It is a subset of linear logic which supports the deployment of denotational semantics methods. Thus we bring that class closer to a machine-independent semantic explanation. And, in the long term, it seems just inevitable that the obsessive denotational semantics quest for a reconstruction of the intensional based only on extensional evidence eventually makes it a powerful tool in complexity.

Surely another potential "application" is in the design of type systems which would embody some sort of static termination proof within an interesting bound. Baillot has recently reported finding a type inference algorithm for elementary linear logic [4]. This is a very interesting, but quite perpendicular issue, bringing in different questions such as programming comfort.

Both the principles of our system and its representation theorem were forecast in Girard's 1995 paper "Light Linear Logic" [8]. Actually the system can be traced further back in time when the first author was trying to build a subsystem of LL with a simplified geometry of interaction interpretation. Without being deadly original our construction improves on the state of the art by giving a new and concise definition of ELL both as a fragment of LL proof nets and in the form of a sequent calculus endowed with a sound cut-elimination procedure (there was a defect in the original formulation as explained in [9]). We also provide a first complete proof of representation, and we do this directly using Kalmar's abstract characterization of elementary recursive functions, in place of Turing machines. And last we develop a novel realizability interpretation of negative intuitionistic LL proofs in λ-calculus with pairing, which is of independent interest and allows, here, for a simple and thorough checking of our ELL programming.

Coming to the organisation of the matter, we are confronted with a rather embarassing design choice. Proof nets offer a graphical syntax for representing LL proofs which is particularly handy for studying the dynamics of proofs (see, for instance, [6]). And ELL can be defined as the subset of proof nets satisfying some *stratification condition*: that any exponential branch with a dereliction leaf (resp. with an axiom leaf) crosses exactly one (resp. zero) !-box. Of course this definition only makes sense after defining proof nets. It is pretty easy to verify then that proof-net normalization preserves the stratification condition. From this it follows that ELL is strongly normalizing. It follows also that the nonadditive fragment is confluent. All formal proofs presented here have been first done using this presentation of ELL, and it would be natural to use this language in the paper too.

However, to cut down preliminaries, and possibly bring this study to a larger audience, we choose to present ELL as a sequent calculus system. For the same reasons we will actually never use more than the *negative intuitionistic* fragment of ELL, denoted by IELL, which is enough to implement elementary recursive functions. Cut-elimination certainly looks less elegant in sequent calculus, but there remains an advantage yet, namely that our realizability is going to be simpler to define and to prove adequate.

So we first define ILL, that is the *negative intuitionistic* fragment of linear logic, in the guise of a sequent calculus; then we proceed to the definition of IELL as a subset of those proofs meeting (some suitable rephrasing of) the stratification condition and provide some discussion of alternative subsystems. Next, we define cut-elimination and draw from there any properties we need in the remainder of the paper. A last preliminary step is the attachment to any ILL proof of a λ-term which will be useful in verifying the programming of the last section. We then turn to the definition of what it means to represent a function in ILL and IELL, and prove the representation theorem.

We thank Thierry Joly for a crash-course on the many different ways elementary time can be defined in classical subrecursion theory.

2. Elementary linear logic

2.1. ILL and IELL as sequent calculi

2.1.1. Statics

Formulas are built over a countable set of propositional variables with the following operators: $\multimap$ the linear implication, & the additive conjunction, ! the exponential, and $\forall$ the second order universal quantifier. Observe that $\multimap$, &, and $\forall$ are all right reversible. This fragment is usually called the negative fragment of intuitionistic linear logic [7]. *Sequents* are of the form $\Gamma \vdash A$ where Γ is a finite multiset of formulas and A is a formula. An ILL proof is a proof tree constructed from the rules below:

Identity group

$$\frac{}{A \vdash A}\,\text{axm} \qquad \frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\,\text{Cut}$$

Logical group

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,\multimap \qquad \multimap\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\,\& \qquad \&1\frac{\Gamma, A \vdash C}{\Gamma, A \& B \vdash C} \qquad \&2\frac{\Gamma, B \vdash C}{\Gamma, A \& B \vdash C}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall X A}\,\forall \quad (X \text{ not free in } \Gamma) \qquad \forall\frac{\Gamma, A[B/X] \vdash C}{\Gamma, \forall X A \vdash C}$$

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,!\text{-box} \qquad \text{der}\frac{\Gamma, A \vdash C}{\Gamma, !A \vdash C}$$

Structural Group

$$\text{w}\frac{\Gamma \vdash C}{\Gamma, !A \vdash C} \qquad \text{ctr}\frac{\Gamma, !A, !A \vdash C}{\Gamma, !A \vdash C}$$

2.1.2. Various subsystems

Structural rules allow for duplication and erasure in cut-elimination. The other two rules managing the exponential, or the "bang" namely dereliction and promotion, are in charge of controlling these duplications. Prohibiting all of them results in a system with lazy normalization in linear time. Elementary linear logic and its cousin light linear logic, representing exactly polytime functions, also are obtained by constraining these rules, but in a less brutal way.

Definition 1. IELL consists in the proofs of ILL satisfying the following stratification condition: any left occurences of an exponential formula $!A$ introduced by a dereliction rule (an axiom rule) has exactly one (zero) descendant in the context of a promotion rule.

Typical sequents that are provable in ILL but *not* provable in IELL are $!X \vdash !!X$ and $!X \vdash X$ for X a propositional variable. Their shortest proofs respectively are

$$\delta = \frac{!X \vdash\ !X}{!X \vdash !!X} \quad \text{and} \quad \epsilon = \frac{X \vdash X}{!X \vdash X},$$

and, indeed, none are in IELL: δ is not because it violates the axiom stratification condition, and ϵ is not either because it violates the dereliction stratification condition. Assuming cut-elimination, it is easy to show that none of these sequents are provable in IELL. These correspond categorically to the natural transformations making the usual LL exponential a comonad:

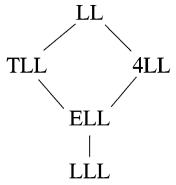
$$X \overset{\epsilon_X}{\leftarrow} !X \overset{\delta_X}{\rightarrow} !!X$$

which brings in the interesting question of whether one can have a categorical description of (what it means to be a model of) elementary linear logic.

Note that the stratification condition runs somewhat against the tradition of sequent calculus by introducing an additional *nonlocal* condition. Not only does one need to check each rule as usual, but in addition, for each dereliction or axiom rule creating a "!" one has to chase down the line of descendants of this formula occurrence and count how many promotion rules it crosses. In particular a subproof of an ELL-proof need not be in ELL itself.

Other meaningful subsystems can be obtained by constraining this crossing-number c in various ways. For instance TLL corresponds to the constraint $c \leqslant 1$ and 4LL to $c \geqslant 1$. So that the former variant admits ϵ and refuses δ, while the latter makes the symmetric choice, they intersect in ELL. Little is known about these variants, except that they are both closed by cut-elimination and TLL is strong enough to encode primitive recursion. And, of course, there is a very interesting smaller system sitting inside ELL, namely Girard's LLL, which is obtained by further constraining the "width" of promotion rules, that is the size of the promotion rules' contexts.

Fragments are named after standard terminology in modal logic (except ELL which should be called KLL). So far the situation is:



but combining width- and crossing-constraints generates many other stable subsystems of linear logic.

2.1.3. Dynamics

We now informally describe the cut-elimination procedure with which we will equip LL. It is the so-called q-protocol, first explained in [5] for classical logic. This is a "big-step" cut-elimination

procedure more in the style of natural deduction normalization than the usual "small-step" cut-elimination.

Let A be an occurrence of a formula in a proof π and define A's *tree* in π to be the subtree of π that contains A's ancestors. Its leaves may be logical rules, axioms, and weakenings introducing A's primeval ancestors. A's tree is said to be *flat* when A is principal in a logical rule.

Now, consider an ILL proof with a cut between two occurrences A_0 and A_1 of the same formula A:

$$\dfrac{\dfrac{\pi_0}{\Gamma \vdash A_0} \quad \dfrac{\pi_1}{A_1, \Gamma' \vdash C}}{\Gamma, \Gamma' \vdash C}.$$

This cut will be handled by one of the three steps described below depending on whether A_0's and A_1's trees are flat. These steps are mutually exclusive and one of them always applies.

1. The *first structural step*, or simply the S1 step, applies when A_0's tree is not flat. It consists in cutting $\frac{\pi_1}{A_1, \Gamma' \vdash C}$ under A_0's logical leaves in π_0, substituting it to axioms $A_0 \vdash A_0$ introducing A_0's leaves in π_0, and replacing A_0's weakening leaves with weakenings on the context formulas Γ' and C, transferring any contraction on A in A_0's tree to contractions on Γ' and C. (Indeed, in the particular case of ILL, A_0's tree is filiform so that it has only one leaf, which must be introduced by a logical rule or an axiom, since there is no right weakening in ILL; nevertheless the q-protocol works in general in a classical ambient calculus).
2. The *second structural step*, or simply the S2 step, applies when A_0's tree is flat and A_1's is not. It consists in cutting $\frac{\pi_0}{\Gamma \vdash A_0}$ under A_1's logical leaves in π_1, substituting π_0 to A_1's axiom leaves, and replacing A_1's weakening leaves with weakenings on the context formulas Γ and transferring any contraction on A in A_1's tree to contractions on Γ. (Again in the particular case of ILL there might only be contractions and weakenings in A_1's tree if A is an exponential formula of the form $!B$ for some B).
3. If no structural steps apply then A_0's and A_1's trees are both flat. Then a *logical step* applies depending on A's principal connective or quantifier. For instance, in the exponential logical rule:

$$\dfrac{\dfrac{!\Gamma \vdash A_0}{!\Gamma \vdash !A_0} \quad \dfrac{A_1, \Gamma' \vdash C}{!A_1, \Gamma' \vdash C}}{\Gamma, \Gamma' \vdash C}$$

reduces to:

$$\dfrac{!\Gamma \vdash A_0 \quad A_1, \Gamma' \vdash C}{!\Gamma, \Gamma' \vdash C}.$$

For a reason to be clear in the next proposition, we modify slightly the general protocol just explained in the particular case where $A = !B$. In this case we ask that the S2 step includes the last reduction just presented (promotion/dereliction) though it is a logical one.

The natural embedding of ILL in proof nets allows us to simulate the q-protocol with proof-net normalization.

In the following we consider only the *lazy* cut-elimination, where one never performs an S2 step when some node of A_1's tree belongs to the context of a right &-rule. That lazy protocol has three properties:

1. it is strongly normalizing;
2. from a recent result of Tortora de Falco [12], it follows that it is also confluent, up to the commutation of structural rules;
3. when applied to proofs of *simple* sequents, i.e., with no positive subformulas of the form $A\&B$ nor any negative subformulas of the form $\forall X\ A$, it leads to cut-free proofs.

Now the expected property:

Proposition 2. IELL *is stable under the q-protocol.*

This is easily checked. An S2 step on an exponential formula is the only nonroutine case. The modification of S2 is required here, as well as the condition that exponential left formulas in axioms have *no* descendants in the context of a promotion rule.

Call the *depth of a rule* in a proof the number of promotion rules below that rule. The key dynamic property of IELL, which the stratification condition directly enforces, is (1) the depth of any rule, different from a cut, is invariant under the q-protocol; and (2) the depth of a cut may only increase under the q-protocol. Thus an IELL proof can be normalized by clearing out cuts in depth-increasing order. Call the *depth of a proof* that of the deepest rule and the *size of a proof* the number of its rules.

Let π be a proof of size s and depth d, if we neglect S1 and L steps which have no effect on the size s; clearing out all cuts at depth d in π takes at most s steps (a crude estimate on the number of cuts !) and result in a proof of size at most s^{s+1} because each S2 step may at most produce s copies of any deeper rule (again an extremely crude estimate). But, almost everywhere $s^{s+1} \leqslant 2^{2^s}$. So if one proceeds in depth-increasing fashion, cut-elimination time will clearly be bounded by an exponential lower $\psi_{2d}(s)$ with $\psi_{n+1}(x) = 2^{\psi_n(x)}$ and $\psi_0(x) = x$.

Theorem 3. IELL *normalizes in elementary time. That is, there is a function* $\theta(x,y)$ *such that* (1) *for all* π *of size* s *and depth* $d, \theta(s,d)$ *bounds the maximum number of cut-elimination steps in* π; *and* (2) $\theta(.,y)$ *is elementary recursive for any* y.

Of course, $\theta(x,x)$ is not elementary recursive. Better estimates for θ than the one above were given by Kanovich in [10], in sequent calculus, and by Baillot and Pedicini in [2], in the geometry of interaction framework.

2.2. ILL and IELL as typed lambda-calculi

We now attach to each ILL proof a typed term of λ-calculus with pairing. In a typing judgment $\Gamma \vdash t : A$ the context Γ is now a partial application from variables to formulas, defined on all free variables of t. The typing judgments are obtained as follows:

$$\frac{}{x : A \vdash x : A}\,\mathsf{dec} \qquad \frac{\Gamma \vdash u : A \quad x : A, \Delta \vdash t : B}{\Gamma, \Delta \vdash t[u/x] : B}\,\mathsf{sub}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \multimap B}\,\mathsf{abs} \qquad \frac{\Gamma \vdash u : A \quad x : B, \Delta \vdash t : C}{\Gamma, y : A \multimap B, \Delta \vdash t[(y)u/x] : C}\,\mathsf{app}$$

$$\frac{\Gamma \vdash t_1 : A \quad \Gamma \vdash t_2 : B}{\Gamma \vdash \langle t_1, t_2 \rangle : A \& B}\,\mathsf{pair}$$

$$\frac{\Gamma, x : A \vdash t : C}{\Gamma, y : A \& B \vdash t[(\mathsf{fst}\ y)/x] : C}\,\mathsf{fst} \qquad \frac{\Gamma, x : B \vdash t : C}{\Gamma, y : A \& B \vdash t[(\mathsf{snd}\ y)/x] : C}\,\mathsf{snd}$$

$$\frac{\Gamma \vdash \ t : A}{\Gamma \vdash \ t : \forall X A}\ (X \text{ not free in } \Gamma) \qquad \frac{\Gamma, x : A[B/X] \vdash t : C}{\Gamma, x : \forall X A \vdash t : C}$$

$$\frac{!\Gamma \vdash t : A}{!\Gamma \vdash t : !A} \qquad \frac{\Gamma, x : A \vdash t : C}{\Gamma, x : !A \vdash t : C}$$

$$\frac{\Gamma \vdash t : C}{\Gamma, x : !A \vdash t : C} \qquad \frac{\Gamma, x_1 : !A, x_2 : !A \vdash t : C}{\Gamma, x : !A \vdash t[x/x_1, x/x_2] : C}$$

Two remarks are in order here. First, the terms attached to a proof are invariant under both quantifier rules, dereliction, and promotion. Second, if $\Gamma \vdash t : A$ is derivable in the system above then $\Gamma^- \vdash t : A^-$ is derivable in system F with products, where A^- is obtained from A by erasing exponentials, replacing $\multimap$ with $\to$ the intuitionistic implication and & with $\times$ the intuitionistic product (the so-called intuitionistic *forgetful* reading of ILL).

We now equip terms with their usual equality generated by the following:

$$(\lambda x, t\ u) \approx t[u/x],\ (\mathsf{fst}\ \langle t_1, t_2 \rangle) \approx t_1 \quad \text{and} \quad (\mathsf{snd}\ \langle t_1, t_2 \rangle) \approx t_2.$$

If π is an ILL proof, it is assigned, up to renaming, a unique term by the typing system above, which we denote by $t(\pi)$. The reason we introduce these terms in the first place is that $t(\pi)$ can be read as a kind of computational semantics of π:

Proposition 4. ILL *normalization is compatible with its term assignment. That is, given an* ILL *proof* π, *if* π *q-normalizes to* π', *then* $t(\pi) \approx t(\pi')$.

Specifically, both structural steps, S1 and S2, leave the attached term invariant, and so does the quantifier step. Additive and multiplicative logical steps induce a rewriting of the attached term using the corresponding rule above. Working in the negative fragment comes into play here: during an S1 step, any rule that the line of ancestors of a given cut formula traverses is rendered as a substitution at the term level and therefore the attached term is invariant under S1. This would no longer be the case in the presence of the additive disjunction $\oplus$, though even this case could be accomodated at the expense of more term equations.

We are done now with proof-theoretic preliminaries. In the next section we set up the notion of implementing a function with a proof in ILL.

3. Implementing functions in IELL

3.1. Representing integers

Put $N = \forall X!(X \multimap X) \multimap !(X \multimap X)$. This is the formula we choose to be the type of integers. Note that the intuitionistic associated type, N^-, is but the usual integer type in system F. In fact our programming is going to run parallel to usual F programming as far as our typing system permits.

For A any formula, let N_A stand for $!(A \multimap A) \multimap !(A \multimap A)$.

We proceed now to the description of normal, or cut free, integers in IELL. In case $n \neq 0$, n is represented by the proof below, denoted ω_n. By convention double inference lines indicate more than one rule in a row.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \vdash X}\quad\dfrac{}{X \vdash X}}{X, X \overset{1}{\multimap} X \vdash X}\multimap}{\vdots}}{X, X \overset{n-1}{\multimap} X, \ldots, X \overset{1}{\multimap} X \vdash X}\multimap}{X, X \overset{n}{\multimap} X, \ldots, X \overset{1}{\multimap} X \vdash X}}{X \overset{n}{\multimap} X, \ldots, X \overset{1}{\multimap} X \vdash X \multimap X}\ \text{ders}}{!(X \overset{n}{\multimap} X), \ldots, !(X \overset{1}{\multimap} X) \vdash !(X \multimap X)}\ \text{!-box}\ \text{ctrs}}{!(X \multimap X) \vdash !(X \multimap X)}\multimap}{\vdash N_X}\forall}{\vdash N}$$

Or, in case $n = 0$, it is represented by the following proof, denoted ω_0:

$$\dfrac{\dfrac{\dfrac{\dfrac{X \vdash X}{\vdash X \multimap X}\ \text{w}}{!(X \multimap X) \vdash X \multimap X}\multimap}{!(X \multimap X) \vdash !(X \multimap X)}\ \text{!-box}}{\vdash N_X}\forall}{\vdash N}$$

Note that ω_{ns} are indeed in IELL, trivially so for ω_0 since it contains no dereliction or exponential axiom and less trivially for other ω_n's since all their derelictions, of which they have exactly n, have one descendant in the unique promotion. All of them are of depth 1 and ω_n's size is $4(n+1)$. Also note that $t(\omega_n) = \lambda f \lambda x.(f(\ldots(fx)))$, i.e., the nth Church numeral, which we shall denote by $\underline{n}$.

Observe also that, up to commutation of weakenings, derelictions, and contractions, these are all cut free proofs of N in IELL. Except for 1, which has two representations. But this is already the case in system F.

3.2. Implementing functions

We need a few notations before we are in position to define what it means to represent a function.

If $\frac{\pi}{A_1,\ldots,A_k\vdash B}$ and $\frac{\pi_1}{\vdash A_i}$, $i=1,\ldots,k$, are proofs in IELL, then $\frac{\pi(\pi_1,\ldots,\pi_k)}{\vdash B}$ will stand for π cut against the π_i's; that is:

$$\dfrac{\dfrac{\pi_1}{\vdash A_1}\quad\dfrac{\pi}{A_1,\ldots,A_k\vdash B}}{A_2,\ldots,A_k\vdash B}\qquad\vdots\qquad\dfrac{\dfrac{\pi_k}{\vdash A_k}\quad A_k\vdash B}{\vdash B}$$

If A is a formula, let $!^pA$ stand for $!\ldots!A$, that is A prefixed by p exponentials. If $\frac{\pi}{\vdash A}$ is a proof in IELL, then $\frac{!^p\pi}{\vdash !^pA}$ will stand for π promoted p times. Note that $!^p\pi$ is still in IELL.

Definition 5. Let f be a k-ary function from integers to integers; we say that f is representable (or programmable) in IELL if there exists an integer $p\geqslant 0$ and an IELL proof $\frac{\pi}{N,\ldots,N\vdash !^pN}$ such that for all $n_1,\ldots,n_k$:

$\pi(\omega_{n_1},\ldots,\omega_{n_k})$ q-normalises to $!^p\omega_{f(n_1,\ldots,n_k)}$.

When $p=0$, we say such a π is *flat*; when not, we say π is *oblique*.

Note that the term $t(\pi)$ attached to such a π is an open term with at most k free variables. (This saves systematically ending proofs with k right implication rules.) As said, it always will be typable of type $N^-,\ldots,N^-\vdash N^-$, in system F with products.

If f is representable then it must be total since all proofs are strongly normalizing. Conversely:

Lemma 6. *Let* $\frac{\pi}{N,\ldots,N\vdash !^pN}$ *be an* IELL *proof*; *then* (1) π *represents a unique total recursive function*, *say* f, and (2) $t(\pi)$ *also represents* f.

Indeed, for all $n_1,\ldots,n_k$, $\pi(\omega_{n_1},\ldots,\omega_{n_k})$ proves a simple conclusion, namely $!^pN$. It therefore normalizes to a *unique* proof of the form $!^p\omega_m$, for some m. Hence π represents the function which produces m on input $n_1,\ldots,n_k$.

For (2) suppose π is unary, just to ease notations. Call x the free variable of $t(\pi)$ and let n be an integer. Then $\pi(\omega_n)$ normalizes to a unique ω_m as said. Now by compatibility, we know that $t(\pi)[\underline{n}/x]\approx\underline{m}$, which means $t(\pi)$ represents (in the usual sense of λ-terms representing functions over Church numerals) the same function as π.

The second point in the lemma is going to be very useful: to check whether a π represents a function, it suffices to prove that $t(\pi)$ does. And it is very illuminating too, because it says exponentials are only instrumental in collecting information about the term (such as an elementary time termination proof in our case), in no way do they express any computational content. And this is nothing specific to IELL, it is true of negative ILL in general.

Call d the depth of such a π, and s its size. Then $\pi(\omega_{n_1},\ldots,\omega_{n_k})$ is of depth $\max(1,d)$, since $\omega_n s$, as said, are of depth 1. It follows that $\pi(\omega_{n_1},\ldots,\omega_{n_k})$ will normalize in less than $\theta(4(n_1+\cdots$

$+n_k) + 5k + s, \max(1, d))$ steps, since ω_n has size $4n + 4$, and that bound is elementary recursive in $n_1, \ldots, n_k$. Hence:

Theorem 7. *Any function from integers to integers representable in* IELL *is elementary recursive.*

Indeed, q-steps can be simulated on a Turing machine with an elementary recursive slow-down (and not less in general because one S2 step might transform a proof of size s in a proof of size s^2). If f is representable, it can be implemented by a Turing machine running in elementary time in the size of the inputs (be they unary as here or binary, as usual, this makes no difference for the class of elementary recursive functions).

The end of the paper is devoted to a careful proof of the converse:

Theorem 8. *Any elementary recursive function from integers to integers is representable in* IELL.

4. Programming in ELL

We use the abstract following characterization due to Kalmar (cf. [11]): elementary functions are contained in any class of functions: (1) containing constants, projections, addition, multiplication, substraction (or equality test) and (2) closed under composition, bounded sums, and bounded products.

All we have to do now is to program in IELL these basic functions and schemes. In passing we will program a few more, mostly variations on iteration of flat π's.

The method we follow each time consists in (1) exhibiting a proof, (2) checking that it belongs to IELL (in other words that it meets the stratification condition), (3) computing the attached term, and (4) checking that it itself represents the desired basic function or scheme (which is enough to show that the proof represents the intended function or scheme, by the lemma above). We leave it to the reader to verify points (2) which is always trivial and (4) which involves regular λ-calculus computations.

Below, proof will mean proof in IELL.

4.1. Addition and successor

Addition is represented by the following flat proof (of depth 1):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{!(X\multimap X)\vdash !(X\multimap X)}\quad \dfrac{\dfrac{\dfrac{}{X\vdash X}\quad \dfrac{}{X\vdash X}}{X\multimap X, X\vdash X}\multimap}{X\multimap X, X\multimap X, X\vdash X}\multimap}{X\multimap X, X\multimap X\vdash X\multimap X}\multimap}{!(X\multimap X), !(X\multimap X)\vdash !(X\multimap X)}\text{ders}}{N_X, !(X\multimap X), !(X\multimap X)\vdash !(X\multimap X)}\text{!-box}\multimap}{N_X, N_X, !(X\multimap X), !(X\multimap X)\vdash !(X\multimap X)}\multimap}{\dfrac{N_X, N_X, !(X\multimap X), !(X\multimap X)\vdash !(X\multimap X)}{N, N, !(X\multimap X), !(X\multimap X)\vdash !(X\multimap X)}\forall}\text{ctr}}{N, N, !(X\multimap X)\vdash !(X\multimap X)}\multimap}{N, N\vdash N_X}\forall}{N, N\vdash N}$$

The term attached to this proof is $\lambda f.\lambda x.((mf)((nf)x))$, the regular addition, but the typing hardly follows the inductive structure of the term ! This is why sequent calculus, or explicit substitution (as in [1]) is needed to present ELL as a typing system.

Cutting this proof against ω_1 in either left-hand side formula gives a proof representing the successor function (both being flat). The attached term will be $\lambda f.\lambda x.((mf)(fx))$ or $\lambda f.\lambda x.(f((nf)x))$ depending on which left-hand side formula is chosen. Again these are the two regular implementations of the successor function over Church numerals.

4.2. Multiplication

Multiplication is represented by the following flat proof (of depth 0):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{!(X\multimap X)\vdash !(X\multimap X)}\quad \multimap\dfrac{\dfrac{}{!(X\multimap X)\vdash !(X\multimap X)}\quad\dfrac{}{!(X\multimap X)\vdash !(X\multimap X)}}{N_X,!(X\multimap X)\vdash !(X\multimap X)}}{N_X,N_X,!(X\multimap X)\vdash !(X\multimap X)}\,\forall}{N,N,!(X\multimap X)\vdash !(X\multimap X)}\,\multimap}{N,N\vdash N_X}\,\forall}{N,N\vdash N}$$

the term attached being $\lambda f.(m(nf))$.

4.3. Two iteration schemes

The following scheme $IT'_{S,E}$ generalizes the traditional iteration scheme. Given two proofs S and E (S is mnemonics for *S*tep, and E for *E*xit) we define $IT'_{S,E}$ to be:

$$\dfrac{\dfrac{!\Gamma\vdash \overset{S}{!(A\multimap A)}\quad !\Delta,\overset{E}{!(A\multimap A)}\vdash C}{!\Gamma,!\Delta,N_A\vdash C}\,\multimap}{!\Gamma,!\Delta,N\vdash C}\,\forall$$

If s is attached to S, e to E, and x is the free variable associated to the left-hand side formula $!(A\multimap A)$, then $e[(n\ s)/x]$ is attached to $IT'_{S,E}$.

Let B (for *B*ase) be a proof of $\Delta\vdash A$ and S^- be a proof of $\Gamma, A\vdash A$. We get a particular case of $IT'_{S,E}$, with $C=!A$, which we denote $IT_{S^-,B}$, by:

$$\dfrac{\dfrac{\dfrac{\overset{S^-}{\Gamma,A\vdash A}}{\Gamma\vdash A\multimap A}\,\multimap}{!\Gamma\vdash !(A\multimap A)}\,\text{ders}\,!\text{-box}\quad \multimap\dfrac{\dfrac{\overset{B}{\Delta\vdash A}\quad A\vdash A}{\Delta,A\multimap A\vdash A}}{!\Delta,!(A\multimap A)\vdash !A}\,\text{ders}\,!\text{-box}}{!\Gamma,!\Delta,N_A\vdash !A}\,\forall}{!\Gamma,!\Delta,N\vdash !A}$$

If s is attached to S^-, b to B, and x is the free variable associated to the left-hand side formula A in S^-, then $((n\lambda x.s)\ b)$ is attached to $IT_{S^-,B}$. So that $IT_{S^-,B}$ implements the traditional iteration scheme. *But*, note that only a *flat* S^- may be iterated.

(In fact, knowing that elementary functions are not closed by iteration, we deduce that no proof of $!N \vdash N$ represents the identity function; otherwise this proof could be used to flatten any oblique π without changing the function represented; there is no way we could think of to prove this directly on the syntax.)

4.4. Composition

4.4.1. Coercions

Remember successor can be represented by a flat proof, call it succ, so that choosing for S^- and B the following components, in $IT_{S^-,B}$:

$$\mathsf{der}\,\frac{\overset{\mathsf{succ}}{N \vdash N}}{!^k N \vdash !^k N}\,!\text{-box} \qquad \frac{\overset{\omega_0}{\vdash N}}{\vdash !^k N}\,!\text{-box}$$

we get, for all k, a proof of $N \vdash !^{k+1}N$, which we call a *coercion.*

These coercions can be used to strip left-hand side exponentiated N's from their exponentials (by cut). The term attached is $((ns)\ \underline{0})$ where s attached to succ and does not depend on k (while the proof does). It is the identity on Church numerals, though it is not equal to the identity. Hence cutting a coercion on a proof π representing f will leave the function represented by $t(\pi)$ invariant, hence that of π, since they are the same. We can therefore freely use them. (In sharp contrast with any proof of $!^{k+1}N \vdash N$ as just discussed.)

4.4.2. Composition

Let $f(x_1,\ldots,x_k)$ and $g(y_1,\ldots,y_i,\ldots,y_m)$ be functions from integers to integers represented by proofs $\frac{\underline{f}}{N,\ldots,N \vdash !^p N}$ and $\frac{\underline{g}}{N,\ldots,N \vdash !^q N}$. Then their linear composition $g(y_1,\ldots,f(x_1,\ldots,x_k),\ldots,y_m)$ can be represented by:

$$\dfrac{\overset{\mathsf{coercions}}{N \vdash !^p N} \qquad \dfrac{\underline{f} \quad \underline{g}}{N,\ldots,N \vdash !^p N \qquad \mathsf{der}\,\dfrac{N,\ldots,N\ldots,N \vdash !^q N}{!^p N,\ldots,!^p N,\ldots,!^p N \vdash !^{p+q} N}\,!\text{-box}}}{N,\ldots,N,\ldots,N,\ldots,N \vdash !^{p+q} N}\,\mathsf{cuts}$$

where the topmost cut is performed on the left-hand side formula corresponding to y_i. Thus we implement linear composition.

To get the full scheme of composition heavier notation is needed. So, suppose we are given a function $g(y_1,\ldots,y_m)$ and m functions $f_i(x_1,\ldots,x_k)$ of the same arity k, respectively represented by g and $\underline{f_i}$'s with obliqueness q and p_i's. Set $p > \max(p_i)$; we first form $!^p g(!^{p-p_i}\underline{f_1},\ldots,!^{p-p_m}\underline{f_m})$, we then contract all left-hand side formulas corresponding to the same x_j, $j = 1,\ldots,k$ (which is always possible, modulo coercions, since $p - p_i > 0$), and finally we use again coercions to strip remaining left exponentials. Thus we get a proof of $N,\ldots,N \vdash !^{q+p}N$ representing the general composition. Note that the composite has obliqueness at least $q + 1 + \max(p_i)$.

4.5. Exponential

To represent the exponential function m^n, we simply iterate (with our second iteration scheme) any flat proof representing the function $f(x) = mx$ (which we get from our programming of the multiplication), taking ω_1 as the base case.

The resulting proof is a proof of $N \vdash !N$ of obliqueness 1 and hence cannot be iterated further, which is natural since it would result in programming a nonelementary function !

By composition, yet, one can program exponential towers of arbitrary fixed height. The higher the tower, the more oblique the proof becomes.

4.6. Predecessor and substraction

We get the predecessor as an instance of our iteration scheme $IT'_{S,E}$.

For the step S we choose:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \vdash X}}{!(X \multimap X), X \vdash X}\,\mathsf{w}}{!(X \multimap X),\ X \& X \vdash X}\,\&_2 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{X \vdash X} \quad \dfrac{}{X \vdash X}}{X \multimap X,\quad X \vdash X}\,\multimap}{!(X \multimap X), X \vdash X}\,\mathsf{der}}{!(X \multimap X), X \& X \vdash X}\,\&_2}{!(X \multimap X),\ X \& X \vdash X \& X}\,\&}{!(X \multimap X) \vdash X \& X \multimap X \& X}\,\multimap}{!(X \multimap X) \vdash !(X \& X \multimap X \& X)}\,!\text{-box}$$

The term attached to this proof is $\lambda z.\langle(\mathsf{snd}\, z), (f(\mathsf{snd}\, z))\rangle$ where f is the free variable corresponding to the left-hand side formula $!(X \multimap X)$.

For exit E we choose:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{X \vdash X} \quad \dfrac{}{X \vdash X}\,\&}{X \vdash X \& X}\,\multimap \qquad \&_1\,\dfrac{\dfrac{}{X \vdash X}}{X \& X \vdash X}}{X \& X \multimap X \& X, X \vdash X}}{X \& X \multimap X \& X \vdash X \multimap X}\,\multimap}{!(X \& X \multimap X \& X) \vdash !(X \multimap X)}\,!\,\mathsf{der}$$

The term attached to this proof is $\lambda x.(\mathsf{fst}(g\langle x, x\rangle))$ where g is the free variable corresponding to the left-hand side formula $!(X \& X \multimap X \& X)$.

Then, by applying the first iteration scheme, we get:

$$\dfrac{\dfrac{\dfrac{IT'_{S,E}}{N, !(X \multimap X) \vdash !(X \multimap X)}\,\multimap}{N \vdash N_X}\,\forall}{N \vdash N}$$

which represents the predecessor. Observe that the proof *is flat*, so we get substraction with obliqueness 1, by the second iteration scheme.

4.7. Bounded sums and products

To finish the proof we have to deal with bounded sums and products. That is given a representable function $F(i,\vec{x})$, we want to represent the function $\sum_{i=0}^{n} F(i,\vec{x})$ and $\prod_{i=0}^{n} F(i,\vec{x})$. In fact we are going to do a bit more than this by implementing both schemes as instances of a more general one, which we may call *binary iteration*. Expressed in the usual language of recursion, given a binary function G, an $n+1$-ary function F, we can define an $n+1$-ary function H by:

$$H(0,\vec{x}) = F(0,\vec{x}) \quad \text{and} \quad H(n+1,\vec{x}) = G(F(n+1,\vec{x}),H(n,\vec{x})).$$

In the particular case where G is the sum or the product, H is indeed the bounded sum or the bounded product of F. Now, if G is representable by a *flat* proof and F is representable at all, we can represent H using again our iteration scheme $IT'_{S,E}$ as we show below. This is enough to end the proof, since we already provided flat representations of addition and multiplication.

So, suppose we are given a proof $\underline{G}$ of $N,N \vdash N$, the binary flat proof to be iterated which can be thought of as the sum or the product, and a proof $\underline{F}$ of Γ, $N \vdash\ !^pN$. The function H above can then be represented by constructing the proof $IT'_{S_{\underline{F},\underline{G}}\,E}$, where $S_{\underline{F},\underline{G}} : !^2\Gamma \vdash\ !(A\multimap A)$ is

$$
\dfrac{
\dfrac{
\mathsf{w}\,\dfrac{\mathsf{de}\,\dfrac{\&_1\,\dfrac{\overset{\mathsf{succ}}{N \vdash N}}{N\&!^pN \vdash N}}{!(N\&!^pN) \vdash N}}{!\Gamma, !(N\&!^pN) \vdash N}
\qquad
\mathsf{ctr}\,\dfrac{\mathsf{de}\,\dfrac{\&_1\,\dfrac{\&_2\,\dfrac{\dfrac{\dfrac{\overset{\mathsf{succ}}{N \vdash N} \quad \overset{\underline{F}}{\Gamma, N \vdash\ !^pN}}{\Gamma, N \vdash\ !^pN}\,\mathsf{cut} \qquad \mathsf{de}\,\dfrac{\overset{\underline{G}}{N,N \vdash N}}{!^pN, !^pN \vdash\ !^pN}\,\text{!-box}}{\Gamma, N, !^pN \vdash\ !^pN}\,\mathsf{cut}}{\Gamma, N, N\&!^pN \vdash\ !^pN}}{\Gamma, N\&!^pN, N\&!^pN \vdash\ !^pN}}{!\Gamma, !(N\&!^pN), !(N\&!^pN) \vdash\ !^pN}}{!\Gamma, !(N\&!^pN) \vdash\ !^pN}
}{
\dfrac{\dfrac{!\Gamma, !(N\&!^pN) \vdash N\&!^pN}{!\Gamma, !(N\&!^pN) \vdash\ !(N\&!^pN)}\,\text{!-box}}{!\Gamma \vdash\ !(N\&!^pN)\multimap!(N\&!^pN)}\,\multimap
}\,\&
}{}
$$

$$
\mathsf{de}\,\dfrac{!\Gamma \vdash\ !(N\&!^pN)\multimap!(N\&!^pN)}{!!\Gamma \vdash\ !(!(N\&!^pN)\multimap!(N\&!^pN))}\,\text{!-box}
$$

and $E : !^2\Gamma, !(A\multimap A) \vdash\ !^{p+2}N$ is

$$
\mathsf{de}\,\dfrac{
\dfrac{
\dfrac{
\dfrac{\mathsf{w}\,\dfrac{\overset{\omega_0}{\vdash N}}{!\Gamma \vdash N} \qquad \mathsf{der}\,\dfrac{\overset{\omega_0}{\vdash N} \quad \overset{\underline{F}}{\Gamma, N \vdash\ !^pN}}{!\Gamma \vdash\ !^pN}\,\mathsf{cut}}{!\Gamma \vdash N\&!^pN}\,\&
}{!\Gamma \vdash\ !(N\&!^pN)}\,\text{!-box}
\qquad
\mathsf{de}\,\dfrac{\&_2\,\dfrac{\overline{!^pN \vdash\ !^pN}}{N\&!^pN \vdash\ !^pN}}{!(N\&!^pN) \vdash\ !^{p+1}N}\,\text{!-box}
}{!\Gamma, !(N\&!^pN)\multimap!(N\&!^pN) \vdash\ !^{p+1}N}\,\multimap
}{!!\Gamma, !(!(N\&!^pN)\multimap!(N\&!^pN)) \vdash\ !^{p+2}N}\,\text{!-box}
$$

and with $IT'(S_{\underline{F},\underline{G}}, E)$ in place, we finally get, by contractions and coercions on $!^2\Gamma$ (Γ is a multiset on N, since all of F parameters are taken to be integers), a proof of Γ, $N \vdash\ !^{p+2}N$.

The term attached to $S_{\underline{F},\underline{G}}$ is

$$\lambda z.\langle (s(\mathsf{fst}\ z)), g[f[(\mathsf{fst}\ (s\ z))/x']/x, (\mathsf{snd}\ z)/y]\rangle$$

where $s, g[x,y]$, and $f[x']$ are the terms respectively associated to succ, $\underline{G}$ and $\underline{F}$, while the term attached to E is $(\mathsf{snd}(h\langle\underline{0}, f[\underline{0}/x']\rangle))$ where h is the variable corresponding to the left-hand side formula in E.

Observe that, in usual recursion theory, there is no way one could unify both schemes since there we do not have a notion of being flat. Which brings up the question of whether one can actually show some other binary flatly representable function (that is other than bilinear polynomials which obviously can be obtained from multiplication and addition): we do not know.

5. Conclusion

One of the outcome of this careful proof of the representation theorem is that the use of second order has been entirely confined to the first iteration scheme. Thus, one can presumably rephrase all this as a linear typing system for an elementary system T and use static denotational semantics (such as Baillot's stratified coherence spaces in [3]) to model the part of IELL put to use in the proof and try to chase after a semantic counterpart of stratification.

Another phenomenon of interest is that all our programming maps back by the forgetful intuitionistic reading to LJQ, a logical intuitionistic system which is now known to relate to call-by-value λ-calculus. There is hope that typing such terms with IELL might shed some light on what it means for a term to be "stratified."

Other patent questions are: What kind of elementary functionals are defined here ? Can this approach be furthered to LLL?

References

[1] A. Asperti, Light affine logic, in: Proceedings of LICS'98, Indiana University, Bloomington, USA, IEEE Press, New York, 1998.
[2] P. Baillot, M. Pedicini, Elementary complexity and geometry of interaction, in: Proceedings of TLCA'99, Lecture Notes in Computer Science, vol. 1581, Springer, Berlin, 1999.
[3] P. Baillot, Stratified coherent spaces: a denotational semantics for light linear logic, in: Communication at the Second Workshop on Implicit Computational Complexity ICC'00, 2000.
[4] P. Baillot, Private communication, 2001.
[5] V. Danos, J.-B. Joinet, H. Schellinx, A new deconstructive logic; linear logic, J. Symbolic Logic 62 (3) (1995) 755–807.
[6] J.-Y. Girard, Linear logic, Theoret. Comput. Sci. 50 (1987) 1–102.
[7] J.-Y. Girard, A new constructive logic: Classical logic, Math. Struct. Comput. Sci. 1 (3) (1991) 255–296.
[8] J.-Y. Girard, Light linear logic, Inform. Comput. (1995) 14.
[9] M.I. Kanovich, M. Okada, A. Scedrov, Phase semantics for light linear logic, Theoret. Comput. Sci., to appear. Extended abstract in: 13th Annual Conference on the Mathematical Foundations of Programming Semantics, Pittsburgh, Pennsylvania, March, 1997, Electronic Notes in Theoretical Computer Science 6 (1997).
[10] M. Kanovich, Communication at the Workshop on Linear Logic and Typed Lambda-Calculus, 5–11th April 1998, CIRM Université de Marseille-Luminy, France, 1998.
[11] H. Rose, E, Sub-Recursion: Functions and Hierarchy, Clarendon Press, Oxford, 1984.
[12] L. Tortora de Falco, Additives of linear logic and normalization-Part I: a (restricted) Church-Rosser property, Theoret. Comput. Sci. 294(3) (2003) 489–524.